Int. J. on Recent Trends in Engineering and Technology, Vol. 10, No. 1, Jan 2014

Performance Analysis of OFDM Transceiver with Folded FFT and LMS Filter

Esai Amudha Vani G[1] and Joseph Gladwin S[2]
[1] PG Scholar, Department of Electronics and Communication, SSN College of Engineering,
Kalavakkam – 603 110, India
g.eav90@gmail.com
[2] Assistant Professor, Department of Electronics and Communication,SSN College of Engineering,
Kalavakkam – 603 110, India
josephs@ssn.edu.in

***Abstract*— This paper presents an Orthogonal Frequency Division Multiplexing (OFDM) transceiver that makes use of a low power Fast Fourier Transform (FFT) along with a Least Mean Square (LMS) filter. The folded FFT is developed via folding transformation and register minimization techniques with real values as inputs which leads to reduction in hardware complexity by exploiting the redundancy present in computing the FFT samples and also the amount of power consumed. A LMS filter is also designed for the purpose of noise removal. The OFDM transceiver with the folded FFT and LMS filter is analyzed in terms of error performance to validate the advantages of less power consumption and hardware utilization when compared to the traditional OFDM system with conventional FFT. The individual components and the entire OFDM system that has been proposed are modeled using Verilog HDL and functionally verified using Xilinx ISIM simulator.**

***Index Terms*— Folded FFT, LMS filter, OFDM, Power consumption, noise variance, MSE.**

I. Introduction

Orthogonal Frequency Division Multiplexing (OFDM) is a technique used for multi-carrier modulation that has densely spaced sub-carriers and has achieved a lot of popularity among the broadband society in the last few years. Multi-carrier modulation is used for data-transmission by dividing a high-bit rate data flow into several parallel low bit-rate data streams and using them to modulate several carriers. Multi-carrier transmission has a number of useful properties such as delay-spread tolerance and spectrum efficiency that support their use in broadband communications. Reliable and high quality communication systems with higher spectral efficiency and high data rate are needed with the increasing application of multimedia services. Thus, the advantages of OFDM has developed it into a popular scheme for wideband digital communication, used in applications such as digital television and audio broadcasting, DSL broadband internet access, wireless networks, and 4G mobile communications.
Fast Fourier Transform (FFT) is extensively used in the area of Digital Signal processing (DSP) like filtering, spectral analysis, etc. in order to calculate the Discrete Fourier Transform (DFT). FFT plays an important role in modern digital communications like digital video broadcasting and OFDM systems. The FFT core is one of the modules having high computational complexity in the physical layer of these communication systems. Different algorithms have been developed to decrease the computational complexity of the FFT, of which Cooley-Tukey radix-2 FFT [1] is very popular. Based on the basic radix-2 FFT approach, many other

DOI: 01.IJRTET.10.1.1357

algorithms and architectures for FFT have been proposed in the literature [2] – [9]. Most of these hardware architectures are not fully utilized and require high hardware complexity. High throughput and low power designs are required to meet the speed and power requirements of high speed digital communications, while keeping the hardware overhead to a minimum. So, a new approach to design folded FFT that has less power consumption and hardware complexity via folding transformation [10], [11] and register minimization techniques [12], [13] is proposed in this paper. A Real valued Fast Fourier Transform (RFFT) architecture is formed based on the different algorithms that have been proposed [14] – [16]. The folded FFT that has been proposed is also a RFFT based architecture since it makes use of real values as its inputs. Also, a Decimation-In-Frequency (DIF) approach is used for deriving the proposed FFT architectures.

Noise is a factor that plays a critical role in affecting the performance of any system. In an OFDM system, the noise sources could be of different types like Weiner phase noise, Rician noise, Additive White Gaussian Noise (AWGN) etc. In order to remove these noise sources, a Finite Impulse Response (FIR) digital filter is used in general for better performance. Here, an adaptive FIR filter based on the LMS algorithm [17], [18] is used in order to remove the noise present inside the OFDM. Also, the folded FFT architecture replaces the conventional FFT architecture inside the OFDM system being proposed. Similarly, a folded IFFT is also used instead of the conventional IFFT architecture as a part of the proposed OFDM. The proposed OFDM is then analyzed in terms of error performance to validate the usage of the folded FFT architecture and the LMS filter.

This paper is organized as follows. The architecture for 2-parallel DIF RFFT based on folding transformation is presented in Section II. Section III gives an overview of the adaptive FIR filter based LMS algorithm. In Section IV, the OFDM system with the proposed FFT architecture and LMS filter is presented. The OFDM system with the proposed FFT architecture and LMS filter is analyzed in terms of error performance in Section V. Section VI concludes the analysis with a brief discussion.

II. RFFT ARCHITECTURES VIA FOLDING TRANSFORMATION

In this section, the folding transformation technique to derive a 2-parallel RFFT architecture for radix-2 DIF FFT is illustrated. It can be extended to other radices in a similar fashion. A radix-2 8 point DIF FFT is represented as a Data Flow Graph (DFG) as shown in Fig. 1. The nodes in the DFG represent tasks or computations. In this case, all the nodes represent the butterfly computations of the radix-2 FFT algorithm. In particular, assume nodes A and B have the multiplier operation on the bottom edge of the butterfly.

To transform the DFG, folding sets are used which is an ordered set of operations carried out by the same functional component. The folding transformation [11] is used on the DFG in Fig. 1 to derive a pipelined architecture. Each folding set contains N entries some of which may be null operations.

N is known as the folding factor that represents the number of computations folded into one functional unit. The operation in the ith position within the folding set (where i goes from 0 to $N-1$) is executed by the functional unit during the time partition i. The term i is the folding order, i.e., the time instance at which the node is listed to be executed in hardware. For example, consider the folding set $A = \{\emptyset, \emptyset, \emptyset, \emptyset, A0, A1, A2, A3\}$ for $N = 8$. The operation A0 belongs to the folding set with the folding order 4. The functional unit executes the operations $A0, A1, A2, A3$ at the respective time instances and will be idle during the null operations [11].

Consider an edge 'e' connecting the nodes 'X' and 'Y' with $w(e)$ delays. $Nm + u$ and $Nm + v$, respectively represent the time units at which m-th iteration of the nodes X and Y are scheduled. u and v are called folding orders that must satisfy $0 < u, v < N-1$. The folding equation is given as

$$D_F(X \rightarrow Y) = Nw(e) - P_X + v - u\,. \qquad (1)$$

where P_X is the number of pipeline stages in the hardware unit that executes node 'X'.

Consider folding of the DFG in Fig. 1 with the folding sets

$A = \{\emptyset, \emptyset, \emptyset, \emptyset, A0, A1, A2, A3\}$

$B = \{B2, B3, \emptyset, \emptyset, \emptyset, \emptyset, B0, B1\}$

$C = \{C1, C2, C3, \emptyset, \emptyset, \emptyset, \emptyset, C4\}$

Assume that the butterfly operations do not have any pipeline stages, i.e., $Pa = Pb = Pc = 0$ [11]. The folded architecture can be derived by writing the folding equation in (1) for all the edges in the DFG. These equations are

$D_F(A0 \rightarrow B0) = 2$ $\quad$ $D_F(B0 \rightarrow C0) = 1$

$D_F(A0 \rightarrow B2) = -4$ $\quad$ $D_F(B0 \rightarrow C1) = -6$

$D_F(A1 \rightarrow B1) = 2$ $\quad$ $D_F(B1 \rightarrow C0) = 0$

$D_F(A1 \rightarrow B3) = -4$ $\quad$ $D_F(B1 \rightarrow C1) = -7$

$D_F(A2 \rightarrow B0) = 0$ $\quad$ $D_F(B2 \rightarrow C2) = 1$

$D_F(A2 \rightarrow B2) = -6$ $\quad$ $D_F(B2 \rightarrow C3) = 2$

$D_F(A3 \rightarrow B1) = 0$ $\quad$ $D_F(B3 \rightarrow C2) = 0$

$D_F(A3 \rightarrow B3) = -6$ $\quad$ $D_F(B3 \rightarrow C3) = 1$.

For example, $D_F\ (A0 \rightarrow B0)\ =\ 2$ means that there is an edge from the butterfly node A to node B in the folded DFG with two delays [11]. For the folded system to be realizable, $D_F(X \rightarrow Y)\ \geq 0$ must hold for all the edges in the DFG. Retiming and/or pipelining can be used to either satisfy this property or determine that the folding sets are not feasible. We can observe the negative delays on some edges [11]. The DFG can be pipelined as shown in Fig. 2 to ensure that folded hardware has non-negative number of delays. The folded delays for the pipelined DFG are given by

$D_F(A0 \rightarrow B0) = 2$ $\quad$ $D_F(B0 \rightarrow C0) = 1$

$D_F(A0 \rightarrow B2) = 4$ $\quad$ $D_F(B0 \rightarrow C1) = 2$

$D_F(A1 \rightarrow B1) = 2$ $\quad$ $D_F(B1 \rightarrow C0) = 0$

$D_F(A1 \rightarrow B3) = 4$ $\quad$ $D_F(B1 \rightarrow C1) = 1$

$D_F(A2 \rightarrow B0) = 0$ $\quad$ $D_F(B2 \rightarrow C2) = 1$

$D_F(A2 \rightarrow B2) = 2$ $\quad$ $D_F(B2 \rightarrow C3) = 2$

$D_F(A3 \rightarrow B1) = 0$ $\quad$ $D_F(B3 \rightarrow C2) = 0$

$D_F(A3 \rightarrow B3) = 2$ $\quad$ $D_F(B3 \rightarrow C3) = 1$.

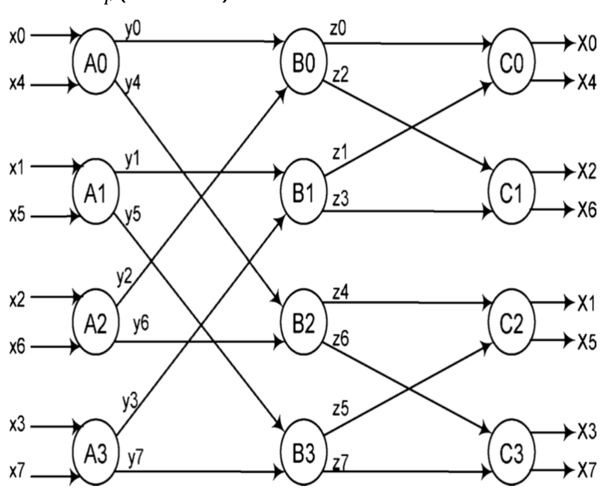

Figure 1. DFG of a radix-2 8-point DIF FFT [11]

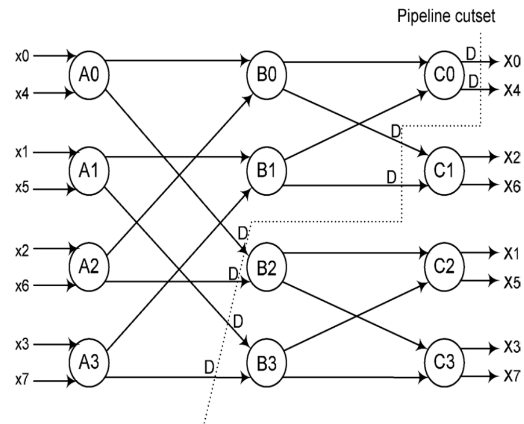

Figure 2. Pipelined DFG of a 8-point DIF FFT as a pre-processing step for folding [11]

It is observed that 24 registers are required to implement the folded architecture [11]. Lifetime analysis technique is used to design the folded architecture with minimum possible registers. For example, in the current 8-point FFT design, consider the variables y0, y1….y3 i.e., the outputs at the nodes $A0, A1, A2, A3$ respectively [11]. It takes 16 registers to synthesize these edges in the folded architecture. The linear lifetime chart for these variables is shown in Fig. 3. From the lifetime chart, it can be seen that the folded architecture requires 4 registers as opposed to 16 registers in a straightforward implementation [11]. The next step is to perform forward-backward register allocation. The allocation table is shown in Fig. 4. Similarly, we can apply lifetime analysis and register allocation techniques for the variables $x0, \dots, x7$ and $z0, \dots., z7$, inputs to the DFG and the outputs from nodes $B0, B1, B2, B3$ respectively, as shown in Fig. 1. From the allocation table in Fig. 4 and the folding delay values, the final architecture in Fig. 5 can be synthesized. The above procedure can be carried out for any N point DIT or DIF FFT in order to form the folded architecture.

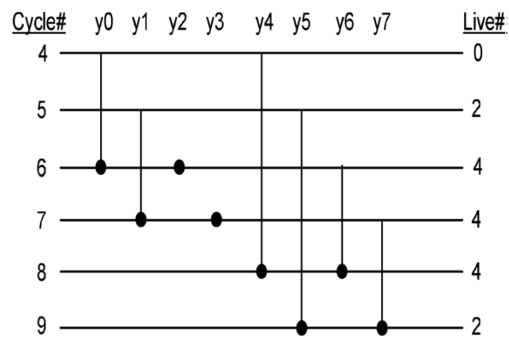

Figure 3. Linear lifetime chart for the variables $y0, y1, y2$ ...[11]

	I/P	R1	R2	R3	R4
4	y0,y4				
5	y1,y5	y4		y0	
6	y2,y6	y5	y4	y1	y0
7	y3,y7	y6	y5	y4	y1
8		y7	y6	y5	y4
9			y7		y5

Figure 4. Register allocation table for the data represented in Figure 3 [11]

A. 16 Point 2-Parallel Radix-2 RFFT Architecture

The procedure described above can be repeated for a 16 point RFFT to obtain the folded architecture. The 2-parallel folded architecture handles two inputs per clock cycle and so two output samples are processed per clock cycle. For RFFT, the input sequence is assumed to be real and it is easy to show that if $x[n]$ is real, then the output $X[k]$ is symmetric, i.e.,

$$X[N - k] = X^{*}[k] . \qquad (2)$$

Using this property, $(N/2) - 1$ outputs can be removed which are redundant and only $N/2 + 1$ outputs of the FFT are needed [11]. Thus the simplified flow graph is obtained by removing the redundant values as shown in Fig. 6. The numbers on the edges indicate the clock cycle numbers in which those intermediate samples are computed [11]. As per the input order of data, the first butterfly computes the even samples and then the odd samples in the following order: (0,8), (2,10), (4,12), (6,14), (1,9), (3,11), (5,13), (7,15).

Consider the folding sets

$$A = \{A0, A2, A4, A6, A1, A3, A5, A7\}$$

$$B = \{B5, B7, B0, B2, B4, B6, B1, B3\}$$

$$C = \{C3, C5, C7, C0, C2, C4, C6, C1\}$$

$$D = \{D2, D4, D6, D1, D3, D5, D7, D0\}$$

The folded architecture is derived by writing the folding equation in (1) for all the edges. Pipelining and retiming are done to get non-negative delays in the folded architecture [11]. After making use of register minimization and forward-backward allocation techniques, the final folded architecture is obtained as shown in Fig.7.

The switch can be replaced by multiplexers that are controlled by values ‘0’ and ‘1’ such that the correct inputs are switched in at the appropriate clock cycles to produce two output samples in one clock cycle. The entire architecture is modeled using Verilog HDL. Here,

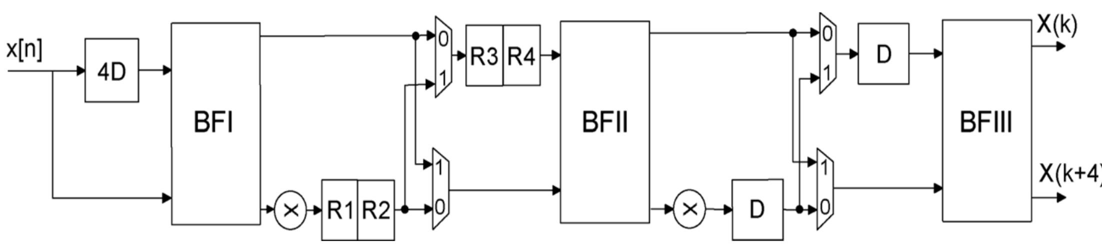

Figure 5. Final folded architecture for 8 point DIF FFT [11]

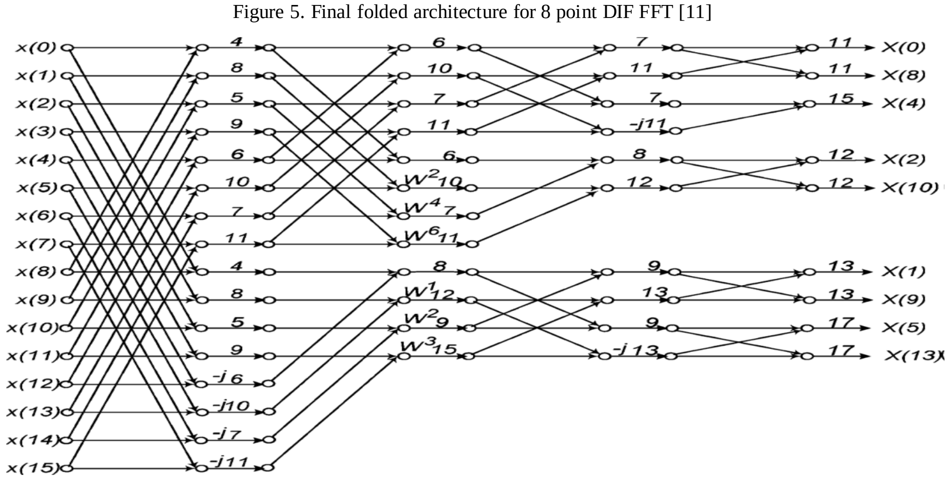

Figure 6. Simplified flow graph of a 16 point radix-2 DIF FFT [11]

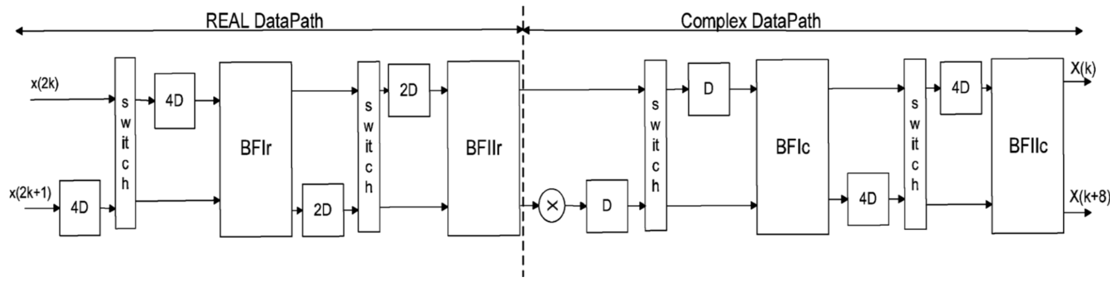

Figure 7. 2- Parallel folded architecture of radix-2 16 point DIF RFFT [11]

the twiddle factors used are considered as floating point numbers which cannot be represented in Verilog HDL and so a 32 bit precision IEEE 754 floating point format is used to represent the twiddle factors for HDL simulation. The twiddle factors are added and multiplied with the inputs and so, the addition and multiplication of the twiddle factors with the respective inputs are accomplished by designing a floating point adder and a floating point multiplier based on the IEEE 754 floating point format.

III. LMS FILTER

Digital filters are a very important part of digital signal processing. Their extra-ordinary performance is one of the key reasons that digital signal processing has become so popular. Digital filters can be used for:

1. Separation of signals that have been combined.
2. Restoration of signals that have been distorted.

A digital filter is a numerical procedure or an algorithm that transforms a given sequence of numbers in to a second sequence that has some desirable properties, such as less noise or distortion. Input sequence to a digital filter can be generated in several ways. If the continuous time signal is denoted by $X(t)$, then the values of the discrete time sequence are denoted as,

$$X(nTs) = X(t) \tag{3}$$

where $t = nTs$ and Ts is the sampling period.

The FIR filter computations play an important role in the DSP systems as the adaptive type. A convolution operation can be given as the weighted summation of the input sequences that is used in the process of FIR filtering. A large amount of multiply-and accumulate operations are present that attracts many researchers to study this issue.

Adaptive digital FIR filters as shown in Fig. 8 are widely employed in practical real-time digital signal processing, communication and networks applications. They utilize different training techniques for updating the filter weights in dynamic environments. By adaptive, we mean a self-designing device that has the following characteristics:

1. It contains a set of adjustable filter co-efficients.
2. The co-efficients are updated in accordance with an algorithm.
3. The algorithm operates with arbitrary initial conditions; each time new samples are received for the input signal and the desired response, appropriate corrections are made to the previous values of the filter coefficients.
4. The adaptation is continued until the operating point of the filter on the error performance surface moves close enough to the minimum points.

Although many adaptive training approaches were introduced for real- time filtering applications, the LMS adaptive algorithm is practically used due to its simplicity and demonstrated efficient performance.

The LMS algorithm was developed by Windrow and Hoff in 1959 [17]. Least mean squares algorithms are used in adaptive filters to identify the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). The least mean squares (LMS) algorithms alter the filter coefficients to reduce the cost function. The LMS algorithms do not involve any matrix operations when compared to Recursive Least Squares (RLS) algorithms. Therefore, the LMS algorithms require fewer computational resources and memory than the RLS algorithms. The execution of the LMS algorithms also is less complex than the RLS algorithms. However, the eigen value spread of the input correlation matrix, or the correlation matrix of the input signal, might affect the convergence speed of the resulting adaptive filter.

The important aspect of the LMS algorithm is its simplicity. It needs neither the calculation of the correlation function, nor the matrix inversion. It utilizes the Mean Square Error (MSE) as a decisive factor. LMS utilizes the step size parameter, input signal and the difference of the needed signal and the filter output signal to repeatedly calculate and update the filter coefficients set.

Most linear adaptive filtering problems can be formulated using Fig. 9. That is, an unknown system is to be recognized and the adaptive filter attempts to adapt the filter to make it as close as possible to, while using only observable signals $x(n)$, $d(n)$ and $e(n)$ [$y(n)$, $v(n)$ and $h(n)$ are not directly observable]. The LMS algorithm iteratively updates the coefficient and feeds it to the FIR filter. The FIR filter then uses the coefficient $h(n)$ along with the input reference signal $x(n)$ to generate the output $y(n)$. The output $y(n)$ is then subtracted from the desired signal $d(n)$ to generate an error, which is used by the LMS algorithm to compute the next set of coefficients.

The convergence time of the LMS algorithm is dependent on the step size μ. If μ is small, then it may take an elongated convergence time and this may beat the intention of using an LMS filter [18]. However if μ is too large, the algorithm may never converge. The selection of the step-size parameter and the order of the filter efficiently concludes about the performance of the LMS algorithm. The LMS algorithm [19] for a p^{th} order algorithm can be summarized as follows:

Parameters: p = filter order

μ = step size

Initialization: $h(0) = 0$

Computation: For $n = 0,1,2,\ldots$

$$x(n) = [x(n), x(n-1), \ldots, x(n-p+1)]^T \quad (4)$$

$$e(n) = d(n) - h^H(n)x(n) \quad (5)$$

$$h(n+1) = h(n) + \mu e^*(n)x(n) \quad (6)$$

There are two main reasons for the LMS adaptive filter to be so popular:

1. It is much easy to implement the adaptive filter in software and hardware due to its computational simplicity and efficient memory use.
2. It performs efficiently in the existence of numerical errors caused by finite-precision arithmetic.

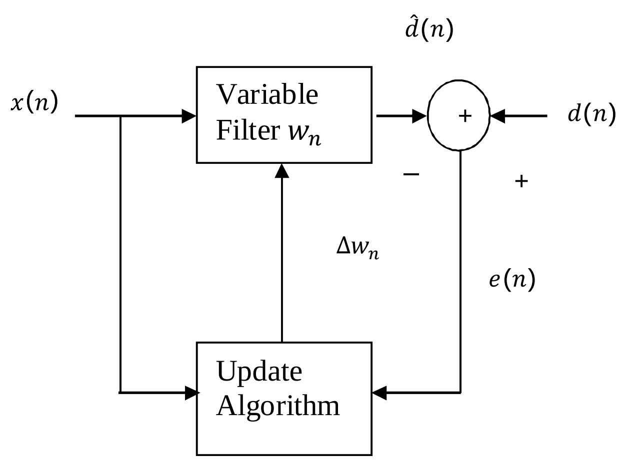

Figure 8. Block diagram of an adaptive filter

IV. Proposed OFDM With Folded FFT And LMS Filter

OFDM is a multi-carrier transmission technique that has been recently recognized as an excellent method for high speed bi-directional wireless data communication. It is also a 'broadband' technique in that the modulated signal fills out a large bandwidth. It is a multi-channel modulation system employing Frequency Division Multiplexing (FDM) of orthogonal sub-carriers, each modulating a low bit-rate digital stream.

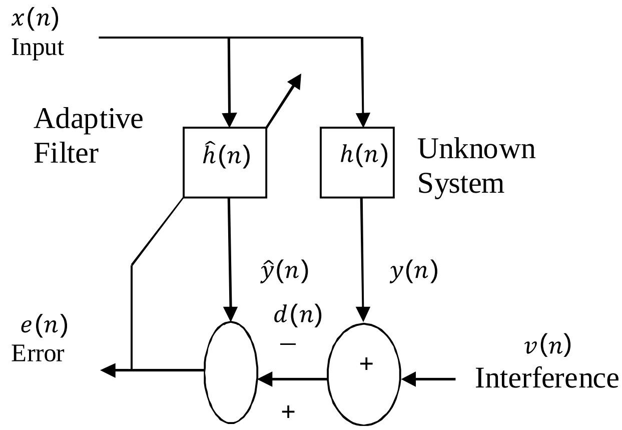

Figure 9. Adaptive filter adjustment using LMS algorithm

The OFDM transmitter usually consists of a filter to remove any unwanted noise that distorts the original input signal along with an Inverse Fast Fourier Transform (IFFT) modulator and a FFT demodulator on the receiver side. In the proposed OFDM transceiver, the folded FFT and IFFT replaces the traditionally used conventional FFT and IFFT blocks respectively and a LMS algorithm based filter removes the noise that gets added with the input signal.

A. Basic Blocks of the Proposed OFDM

The proposed OFDM shown in Fig. 10 is formed by slightly modifying the blocks of a conventional OFDM. The following are the blocks used in the proposed OFDM:

1. Input Data.
2. Line Coding.
3. Serial to Parallel.
4. Block modulator.
5. Parallel to Serial.
6. Filter
7. Serial to Parallel.
8. Block Demodulator.
9. Parallel to Serial.
10. Output Data.

Usually, an image or an audio acts as the input data source for an OFDM. Here, an image of size 1*64 pixels is used as the input data. The image is read using MatLab and each value of the image is converted to IEEE 754 single precision format. This is done as the folded FFT has earlier been implemented in IEEE 754 single precision format which forms a part of the OFDM transceiver and it would facilitate processing of the other blocks. The values of the image that were read using MatLab are later used in Xilinx for the purpose of simulation.

Line Coding generally involves the process of symbol to signal mapping where the values of the input data are mapped to high and low signal levels, for e.g., binary value of 0 and 1 in the image being mapped to -1 and +1 respectively. However, in this case, line coding is carried out as binary representation of the image pixel values. The equivalent binary value in IEEE 754 single precision format of the image undergoes the process of line coding where a value of ‘0’ is assigned as 0V and a value of ‘1’ is assigned as 1V. This is used as the input for the next block.

The 64 pixel values undergo a serial to parallel conversion depending upon the size of FFT and IFFT used inside the OFDM. Here, a 16 point IFFT and FFT are used. So, 64/16 = 4 sets of 16 values each is used in parallel. This can be illustrated as in Fig. 11.

The IFFT acts as the block modulator which is used to convert the signals from frequency domain to time domain. The four parallel sets of 16 values each obtained from the serial to parallel converter are given as input to the block modulator. The IFFT used here is in the folded form as shown in Fig. 12 rather than the

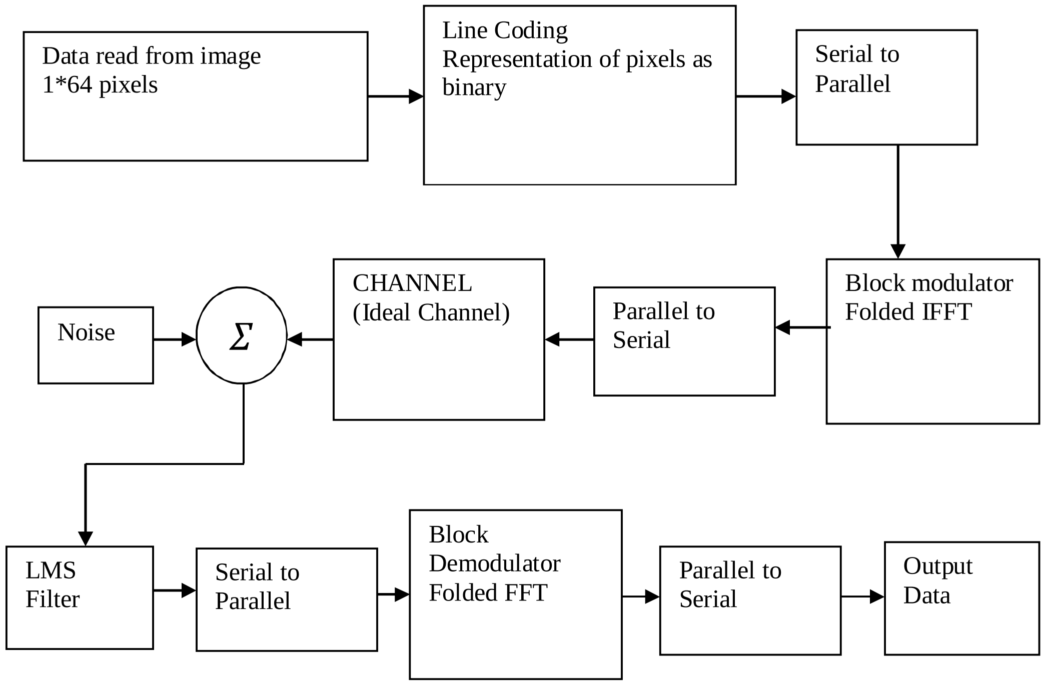

Figure 10. Block Diagram of proposed OFDM

conventional IFFT used traditionally. In the folded IFFT, one butterfly is reused in each stage thus reducing complexity and power consumption when compared to the conventional IFFT.

The four sets of modulated values from the block modulator are given as input to a parallel to serial converter as shown in Fig. 13. The output of this block is 64 serial values. With this block, the OFDM transmitter comes to an end after which the data is passed through a channel and further on to the receiver.

In an OFDM, there are different types of channels used like the Rayleigh Fading channel. In any channel, there will always be a certain loss or distortion of the input data given to the transmitter. Here, the channel used is considered to be ideal which means that there is no loss or distortion of the input signal and all data that is given to the transmitter appears at the receiver.

There are different types of noise sources that are present in an OFDM like the Weiner phase noise, complex rician noise, white Gaussian noise etc. In the proposed model, Additive White Gaussian Noise (AWGN) is used as the noise source. Accordingly, 64 noise values are generated for a mean value of ‘0’ and different values of variance like 0.1, 0.2, 0.3, 0.4 and 0.5 using MatLab and are later used in Xilinx. These 64 noise values are added to the 64 modulated values coming out in series from the end of the channel.

For the process of filtering, a FIR filter is generally used in OFDM. The design of this FIR filter depends upon the value of the coefficients. The selection of the coefficient values can be done using different FIR filter algorithms like LMS, RLMS, Kalman, MSME etc. In the proposed method, the LMS algorithm based FIR filter is used where the coefficients are updated until the mean square error is minimized. The values obtained after the addition of noise and the modulated signal is multiplied with the 64 values of coefficients generated from MatLab for different values of variance.

The signal values obtained after filtering comes out one by one i.e., in series which has to be converted to four sets of 16 values each i.e., in parallel. This facilitates the process of demodulation or FFT.

The block demodulator is the same as that of the block modulator except that the FFT replaces the IFFT. Further, the block modulator is present in the transmitter side whereas the block demodulator is present on the receiver side. The main purpose of using FFT is to convert the signals from time domain to frequency domain. The four parallel sets of 16 values each obtained from the serial to parallel converter are given as input to the block demodulator. The FFT used here is in the folded form rather than the conventional FFT used traditionally. In the folded FFT, one butterfly is reused in each stage thus reducing complexity and power consumption when compared to the conventional FFT.

The four sets of modulated values from the block demodulator are given as input to a parallel to serial converter. The output of this block is 64 serial values. With this block, the receiver part of the OFDM comes to an end.

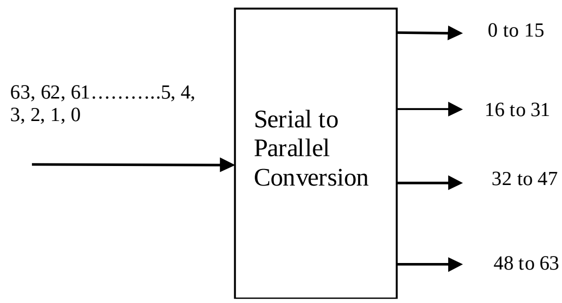

Figure 11. Serial to parallel conversion of 64 image values after line coding

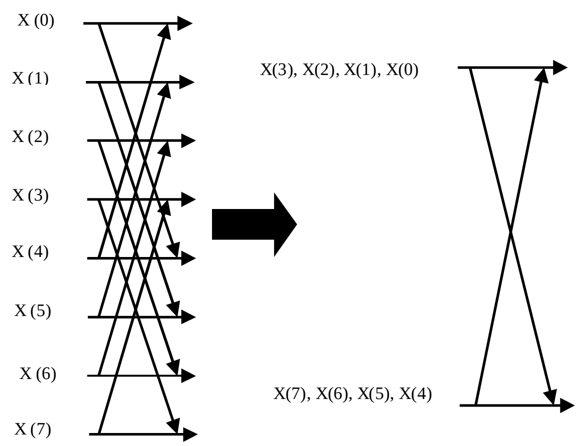

Figure 12. A single stage of conventional IFFT and its corresponding folded IFFT

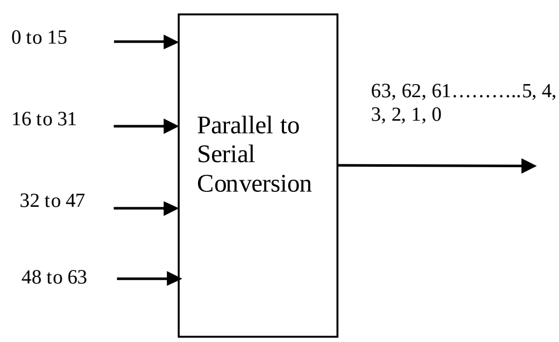

Figure 13. Parallel to Serial conversion of modulated image values

V. Results And Analysis

The simulation results of the OFDM transceiver with the folded FFT and IFFT along with the LMS filter is obtained after modeling the system using Verilog. Based on the simulation results, the performance of the system in terms of error is also validated. In order to facilitate the use of validated components inside the OFDM, the folded FFT architecture that is modeled using Verilog is first discussed besides the LMS filter that is designed using MatLab.

A. Folded FFT architecture and Power Consumption

The simulation was carried out using Xilinx ISIM simulator after a Verilog HDL code was written for both the conventional FFT and proposed folded FFT architectures. The simulation output of the proposed architecture is as shown in Fig. 14. The output values are shown in hexadecimal format for easy manipulation instead of representing it in 32-bit precision IEEE 754 format.

The simulation in Fig. 14 shows the output values d0 [63:0] and d1 [63:0] obtained from the final stage of the proposed folded architecture at the respective clock cycles s[4:0] as shown in Fig. 6. The real part and imaginary part output values are represented with bits 0 to 31 and 32 to 63 respectively. The output that is obtained for input values 0 to 15 is compared with that of the conventional architecture and is found to be the same but with a less utilization of resources and power consumption. The utilization of resources of both the architectures is compared from the synthesis reports obtained. Table I shows the different resources utilized by both the architectures targeting a Virtex -7 device.

The amount of power consumed by the conventional and the folded architectures is estimated by making use of Xilinx XPower Estimator Spreadsheet that targets a Virtex-7 device. A comparison of the estimated power is made between both the architectures which is shown in Table II. It can be found that the proposed folded FFT architecture consumes less than 50% power when compared to the conventional architecture.

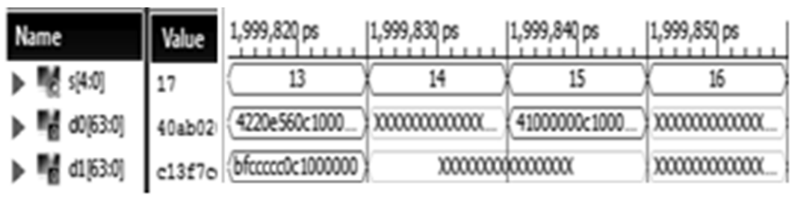

Figure 14. Simulation output for 16- point folded architecture

TABLE I. COMPARISON OF RESOURCES UTILIZED BY THE CONVENTIONAL AND FOLDED FFT ARCHITECTURES

Architecture	#Slice Registers	#Slice LUTs	#Bonded IOBs	# DSP48 E1s
Conventional 16 point FFT	-	71902	2048	54
Folded 2-parallel 16 point RFFT	1319	29174	1153	32

TABLE II. POWER COMPARISON BETWEEN CONVENTIONAL AND FOLDED FFT ARCHITECTURES

Architecture	Estimated Power (Watts)
Conventional 16 point FFT	1.636W
Folded 2-parallel 16 point FFT	0.822W

B. LMS Filter

The LMS filter is designed in Matlab to remove the noise that is added to the input signal. The input signal is an image of size 1*64 pixels which is read using the imread(). Additive white Gaussian noise is generated using randn() with a mean value of ‘0’ and values of variance as 0.1, 0.2, 0.3, 0.4 and 0.5. The LMS filter is then designed using the equations [19] that represent the update of coefficients and error calculation for different iterations. The filter is designed for a particular step size such that the error converges quickly.

C. Proposed OFDM Transceiver

The proposed OFDM is designed with an image of 1*64 pixels acting as the input whose values are obtained from MatLab. Apart from the input, the values of white noise and coefficients of the filter are also read from MatLab. By making use of these values, the entire OFDM transceiver is modeled using Verilog HDL and simulated using Xilinx ISIM simulator.

D. Error Performance

The error performance of the OFDM transceiver depends upon the relation between the variance and the mean square error. Accordingly, as the value of noise variance increases, the value of mean square error would also increase i.e., the mean square error is directly proportional to noise variance.

This error performance is carried out by calculating the error difference between the input data given to the transmitter and the output data obtained from the receiver of the OFDM. Since the noise is generated for different values of variance, the error difference between the input and the output is also calculated for those different values of variance. The Mean Square Error (MSE) of the different values of error is also calculated. Fig. 15 shows the simulation result obtained for a noise variance of 0.5.

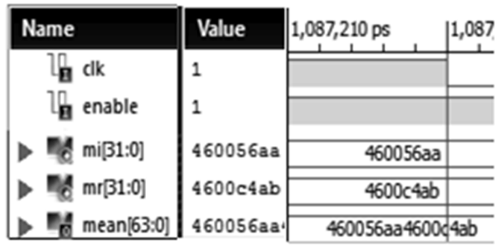

Figure 15. Error value for a variance of 0.5

In the simulation, mr represents the value of mean square error obtained for noise variance value of 0.5 with the clock and enable being set at a high value of '1'.

Table III shows the mean square error values obtained for different values of noise variance. It can be found that the value of mean square error increases as the value of noise variance increases. This satisfies the condition that the mean square error should be directly proportional to the noise variance.

TABLE III. MEAN SQUARE ERROR VALUES OBTAINED FOR DIFFERENT VALUES OF NOISE VARIANCE

Noise Variance	Mean Square Error (MSE)
0.1	39.715706
0.2	41.25567
0.3	42.631386
0.4	43.922195
0.5	45.158844

VI. CONCLUSION

A formal method to derive FFT architectures via folding transformation was proposed. This approach of deriving folded architectures can be carried out for various radices, various levels of parallelism and for either the DIT or the DIF flow graphs. The amount of power consumed can be decreased up to 50% in the folded architectures when compared to the conventional architectures when real values are given as inputs to the FFT. Thus, these low power folded RFFT architectures can be used in applications that require less power consumption and area. So, it was used as a part of an OFDM transceiver along with a LMS filter and its performance in terms of error was analyzed. Since the folded architectures consume less power, it can also be concluded that the OFDM system with this folded architecture would also consume less power when compared with an OFDM system with conventional FFT architecture.

REFERENCES

[1] J. W. Cooley and J. Tukey, "An algorithm for machine calculation of complex fourier series," Math. Comput., vol. 19, pp. 297–301, Apr.1965.

[2] A. V. Oppenheim, R.W. Schafer, and J.R.Buck, Discrete- Time Signal Processing, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[3] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," IEEE Trans. Acoust., Speech, Signal Process., vol. 34, no. 2, pp. 285–295, Apr. 1986.

[4] S. He and M. Torkelson, "A new approach to pipeline FFT processor,"in Proc. of IPPS, 1996, pp. 766–770.
[5] Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," IEEE Journal of Solid state Circuits, vol. 40, no. 8, pp.1726–1735, Aug. 2005.
[6] J. Lee, H. Lee, S. I. Cho, and S. S. Choi, "A High-Speed two parallel radix- FFT/IFFT processor for MB-OFDM UWB systems," in Proc.IEEE Int. Symp. Circuits Syst., 2006, pp. 4719–4722.
[7] J. Palmer and B. Nelson, "A parallel FFT architecture for FPGAs,"Lecture Notes Comput. Sci., vol. 3203, pp. 948–953, 2004.
[8] M. Shin and H. Lee, "A high-speed four parallel radix- 2^4 FFT/IFFT processor for UWB applications," in Proc. IEEE ISCAS, 2008, pp. 960–963.
[9] M. Garrido, "Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time," Ph.D. dissertation, Dept. Signal, Syst., Radio commun., Univ. Politecnica Madrid, Madrid, Spain, 2009.
[10] K. K. Parhi, C. Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," IEEE Journal of Solid state Circuits, vol. 27, no. 1, pp. 29–43, Jan. 1992.
[11] M. Ayinala, M. Brown and K. K. Parhi. " Pipelined Parallel FFT Architectures via folding transformation," IEEE Trans. Very Large Scale Integration Systems, Vol. 20, no. 6, June 2012.
[12] K. K. Parhi, "Systematic synthesis of DSP data format converters using lifetime analysis and forward-backward register allocation," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 39, no. 7, pp. 423–440, Jul. 1992.
[13] K. K. Parhi, "Calculation of minimum number of registers in arbitrary life time chart," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 41, no.6, pp. 434–436, Jun. 1995.
[14] R. Storn, "A novel radix-2 pipeline architecture for the computation of the DFT," in Proc. IEEE ISCAS, 1988, pp. 1899–1902.
[15] Y. Wu, "New FFT structures based on the Bruun algorithm," IEEE Trans. Acoust., Speech Signal Process., vol. 38, no. 1, pp. 188–191, Jan. 1990.
[16] B. R. Sekhar and K. M. M. Prabhu, "Radix-2 decimation in frequency algorithm for the computation of the real-valued FFT," IEEE Trans. Signal Process., vol. 47, no. 4, pp. 1181–1184, Apr. 1999.
[17] J. S. Mason and N. N. Chit, " New approach to the design of FIR digital filters," IEEE Proceedings, Vol. 134, no. 4, pp. 167-180, Aug. 1987.
[18] Asit Kumar Subudhi, Biswajit Mishra and Mihir Narayan Mohanty, "VLSI design and implementation for adaptive filter using LMS filter," International Journal of Computer & Communication Technology, vol. 2, issue 6, 2011.
[19] Monson H. Hayes, Statistical Digital Signal Processing and Modeling, 2nd ed., John Wiley & Sons: 1996.